// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract EventOrganizer {

address public organizer;

uint public ticketPrice;

uint public totalTickets;

uint public ticketsSold;

bool public eventCancelled;

struct EventDetails {

string eventName;

string eventDate;

}

EventDetails public eventDetails;

mapping(address => uint) public ticketHolders;

// Events

event TicketsPurchased(address buyer, uint quantity);

event EventCancelled();

event RefundIssued(address participant, uint amount);

// Modifier to restrict certain functions to the organizer

modifier onlyOrganizer() {

require(msg.sender == organizer, "Only the organizer can perform this action");

}

// Modifier to check if the event is still active

modifier eventActive() {

require(!eventCancelled, "Event has been cancelled");

}

// Constructor to initialize the event

constructor(string memory \_eventName, string memory \_eventDate, uint \_ticketPrice, uint \_totalTickets) {

organizer = msg.sender;

eventDetails = EventDetails(\_eventName, \_eventDate);

ticketPrice = \_ticketPrice;

totalTickets = \_totalTickets;

eventCancelled = false;

}

// Function to purchase tickets

function purchaseTickets(uint \_quantity) public payable eventActive {

require(\_quantity > 0, "You must purchase at least one ticket");

require(msg.value == \_quantity \* ticketPrice, "Incorrect Ether amount sent");

require(ticketsSold + \_quantity <= totalTickets, "Not enough tickets available");

ticketHolders[msg.sender] += \_quantity;

ticketsSold += \_quantity;

emit TicketsPurchased(msg.sender, \_quantity);

}

// Function to cancel the event and allow refunds

function cancelEvent() public onlyOrganizer {

eventCancelled = true;

emit EventCancelled();

}

// Function to claim refund if the event is cancelled

function requestRefund() public {

require(eventCancelled, "Event is not cancelled");

require(ticketHolders[msg.sender] > 0, "No tickets purchased");

uint refundAmount = ticketHolders[msg.sender] \* ticketPrice;

ticketHolders[msg.sender] = 0;

payable(msg.sender).transfer(refundAmount);

emit RefundIssued(msg.sender, refundAmount);

}

// Function to withdraw funds by the organizer (if event is not cancelled)

function withdrawFunds() public onlyOrganizer {

require(!eventCancelled, "Cannot withdraw funds, event is cancelled");

require(address(this).balance > 0, "No funds to withdraw");

payable(organizer).transfer(address(this).balance);

}

// Get the contract's balance (total funds from ticket sales)

function getBalance() public view returns (uint) {

return address(this).balance;

}

}

A screenshot of a computer

Description automatically generated